US Department of Education

Federal Student Aid

June 11, 2007

Architectural Models Phase I

Application Architectural Model v2

Final with Post Delivery Edits

START HERE GO FURTHER FEDERAL STUDENT AID

Table of Contents

Table of Figures

Table of Tables

Section 1: Introduction

This document presents an Architectural Model that documents and communicates Federal Student Aid's architectural vision. There are number of architecture models each covering a particular solution domain (i.e., architectural area). This document specifically addresses application architecture, which provides insight into Federal Student Aid's vision for building business applications. Future architecture model documents will cover architectural areas such as portal and security architecture. This document also provides background on Federal Student Aid enterprise initiatives to achieve the Target State Vision (TSV).

1.1 Statement of Purpose / Objectives

Federal Student Aid plans to initiate numerous acquisitions to implement its TSV. An architecture model is a vehicle for Federal Student Aid to clearly communicate expectations of how solutions offered by vendors should meet their architectural requirements and vision. The architectural model provides a framework within which solutions will operate.

Federal Student Aid's current technical environment is documented by a number of documents developed by Federal Student Aid Chief Information Officer (CIO), Integration Team and their contractors over time. A significant challenge for Federal Student Aid is to specify which one of these documents is required to support the definition of a specific technical solution, and which documents should be included in an acquisition package. Therefore, Federal Student Aid has decided to adopt a standard template, an architectural model that documents Federal Student Aid's architectural expectations. Federal Student Aid will require vendors to use this architectural model to better understand Federal Student Aid's technical environment, available resources, and the standards to which they must adhere. The objective of this standardization of technical architecture documentation is to obtain proposed solutions from vendors that are of higher quality, in compliance with Federal Student Aid standards, and compatible with the Federal Student Aid architectural vision.

1.2 Intended Audience / Usage

This document is intended to be used by a technical audience, such as the Federal Student Aid CIO organization, contractors actively designing /developing a solution and contractors preparing to respond to a Federal Student Aid IT solicitation. This document defines the standards, guidelines, and constraints of Federal Student Aid's application architecture. Vendors will be required to take into account this architectural vision when designing and proposing technology solutions for deployment at Federal Student Aid.

1.3 Scope

Each architectural model is intended to cover relevant architectural elements for specific architectural areas as part of an enterprise-wide solution. Separate documents will be produced for each architectural area and together the series of documents will form a complete reference architecture. Federal Student Aid is currently working on standardizing its programming model and development platform. This document describes the relevant architectural elements of Federal Student Aid's proposed programming model and development platform for building its application architecture.

1.4 Document Organization

This document is organized into the following sections:

- **Section 1: Introduction -** Provides the background and purpose, and scope definition made in the development of Federal Student Aid's architectural model;
- **Section 2: Enterprise Vision -** Provides background information on Federal Student Aid's TSV, establishes the enterprise-level architectural principles followed in the development of the model and introduces the architectural model at a high level;
- **Section 3: Application Architecture -** Presents the application architectural model along with detailed descriptions of the architectural tiers and key technologies leveraged within each tier;
- **Section 4: Architectural Decisions -** Documents key architectural decisions made by the Federal Student Aid's application architecture team. These decisions are presented with associated rationale, alternatives considered and implications of each decision.
- **Section 5: Glossary and Standards -** Identifies and defines key standards and technologies described within the application architecture model.
- **Section 6: Use Cases -** Presents the dynamic view of the application architecture model by walking through a generic task that the application architecture will support.
- **Appendix A: Acronyms -** Defines all acronyms used throughout the application architectural model.

Section 2: Architecture Vision

Architectural models are intended to facilitate successful solutions for building the target state. Architectural models accomplish this by providing a reference that both Federal Student Aid and the solution provider can use as a baseline for communication and architecting of a solution. This section begins with a discussion about the Federal Student Aid TSV which will provide context for the architecture vision. The remainder of this section describes the architecture vision.

2.1 Target State Vision

Federal Student Aid, an office of the U.S. Department of Education, administers Federal student financial assistance programs authorized under Title IV of the Higher Education Act (HEA) of 1965, as amended. Its mission is to ensure that all eligible individuals can benefit from federally funded or federally guaranteed financial assistance for education beyond high school. Federal Student Aid accomplished this mission by offering numerous assistance programs that are funded by public and private sources. There are approximately 13 million students that apply for financial aid each year.

Federal Student Aid has embarked on a major business and systems reengineering effort to create an integrated suite of solutions under the Performance Based Organization (PBO) legislation of 1998. Incorporated within this effort, a TSV has been defined to describe how Federal Student Aid should operate and administer Title IV programs.

To achieve the TSV, Federal Student Aid intends to hire system integrators to help build the target technical environment. A combination of several architectural models will facilitate offerors' understanding of Federal Student Aid's overall architectural vision.

The TSV is based upon the following business goals: 1) delivering student aid in an efficient and cost-effective manner; 2) providing the best access to customers; and 3) maintaining appropriate levels of oversight. Federal Student Aid has identified the following four objectives in order to achieve these goals:

- **Integrate and reengineer business processes to improve operational efficiency and effectiveness, and provide a better experience for those who interact with Federal Student Aid.**

  Federal Student Aid will enhance customers' experience by improving the efficiency and effectiveness of business processes. Target state business processes are logically grouped so that they can leverage common process steps and underlying data. Complex business processes, such as those involving the origination and servicing of aid, will be streamlined, and customers will have the ability to view and manage their aid portfolio from a single access point.

- **Improve data quality and integrity to provide enhanced analytics and reporting capabilities.**

  Federal Student Aid has established both the Enterprise Data Management initiative and the Information Framework (IF) initiative to provide the data services needed to support the target state business processes. The Enterprise Data Management initiative will establish the policies, processes and procedures to ensure that Federal Student Aid focuses on data as an asset. The IF will coordinate the cleansing, movement, and integrity of data to provide consistent enterprise-wide data to all customers and business stakeholders, while minimizing data redundancy and improving data integrity. The IF integrates organizational data (schools, lenders, servicers, and guaranty agencies), person data (applicants, students, and borrowers) and aid data (loans and grants), and provides a host of functional, reporting, and analytical capabilities. Federal Student Aid will have a central point to view and utilize integrated data from multiple systems.

- **Integrate and reengineer information systems to enable target business processes.**

  Federal Student Aid will reengineer and integrate systems to enable the target state business processes. For example, the Integrated Partner Management (IPM) initiative will enable Federal Student Aid to consistently and efficiently manage compliance and oversight for all trading partners. Federal Student Aid will continue reengineering systems to implement other target business processes, such as application processing and servicing, to improve the efficiency and effectiveness of those processes.

- **Implement an integrated, standards-based technical infrastructure that emphasizes enterprise reusable shared assets.**

  Federal Student Aid has adopted a Service Oriented Architecture (SOA) approach to support implementation of the TSV. SOA enables a collection of standard reusable services that will be shared across all of Federal Student Aid. The TSV technical infrastructure includes the following components:

  - *Portal:* A single access point for online information and services required by Federal Student Aid customers, partners, and the general public;
  - *Enterprise Service Bus:* Provides messaging and integration services to facilitate the use of shared functions and access to shared data through SOA;
  - *Security Architecture:* Provides enterprise security services including access and identity management tools;
  - *Gateway:* Provides a single access point for external partner systems to interact with Federal Student Aid systems and services.

2.2 Architectural Principles

Federal Student Aid must collaborate with numerous financial institutions, schools, and government agencies in order to meet the needs of students who qualify for financial aid. This collaboration requires Federal Student Aid to exchange and process large amounts of data each day. Most of Federal Student Aid's applications were developed using proprietary technologies based on legacy platforms and are becoming or already have become technologically obsolete. In addition, many of the current systems at Federal Student Aid are stove-piped, which has resulted in duplication of effort, functionality and data. Consequently it is becoming increasingly difficult to keep these systems up to date, as the overall talent pool for these obsolete technologies is dwindling and the systems do not conform to modern software development standards and architectures.

Therefore, in order to guide the development of future technological environments and to craft an architectural vision that mitigates some of these environmental deficiencies, Federal Student Aid is adopting the following key architectural principles:

Principle 1: The architecture should facilitate reuse of existing IT assets through creation of modular, component-based systems that conform to a service-oriented architecture;

Principle 2: The architecture should facilitate interoperability of systems through the use of standards based technologies, including SOA;

Principle 3: The architecture should facilitate scalability both vertically and horizontally;

Principle 4: The architecture should facilitate high availability through the leveraging of enterprise-class hardware and software.

Principle 5: Service interfaces will be strongly typed.

2.3 Architectural Areas

In order to realize the TSV, Federal Student Aid must solve technical problems in several areas, ranging from the delivery of timely, accurate and relevant business information in a user-friendly manner to the loading and processing of data in large batches. To this end, technical capabilities have been identified to provide solutions in these various problem domains in an overall Reference Architecture.

Each solution domain, or architectural area, is documented through its own architectural model in order to communicate Federal Student Aid's vision for each area. These models provide clarity and insight into how a technical problem can be solved within each of these domains. The Federal Student Aid reference architecture is presented in Figure 2-1 below:

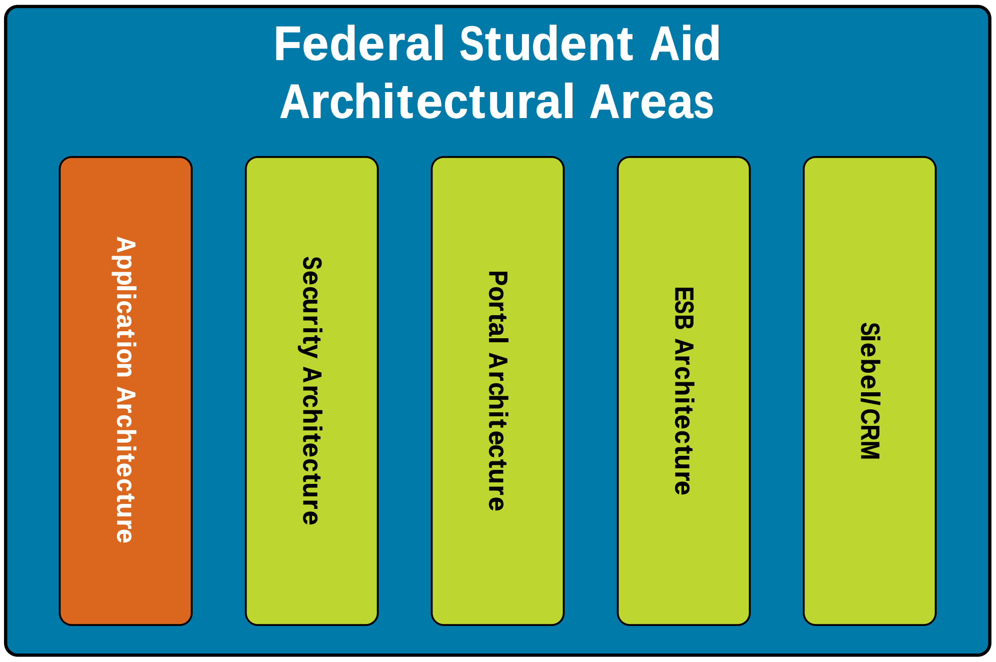

Figure 2-1: Architectural Areas

The following bullets describe each architectural area to be addressed by the reference architecture:

- **Application Architecture** presents Federal Student Aid's vision to enable the capture of business logic within a set of enterprise services, business components and data stores to perform transactional and batch data processing operations.

- **Security Architecture** presents Federal Student Aid's vision to enable secure access to all information technology assets.

- **Portal Architecture** presents Federal Student Aid's architectural vision for crafting solutions that deliver information through enterprise portal technologies.

- **Enterprise Service Bus (ESB) Architecture** presents Federal Student Aid's architectural vision for service integration capabilities to coordinate flow of information between software services and applications.

- **Siebel/Customer Relationship Management (CRM)** presents Federal Student Aid's architectural vision for enabling legacy and Commercial Off The Shelf (COTS) capabilities to coordinate flow of information between new and existing systems.

2.4 Key Concepts

Federal Student Aid's overall approach to application architecture is governed by the following key concepts that will be incorporated into future technology solutions:

- **Service-Oriented Architecture:** SOA is a standards-based architectural philosophy supported by numerous technological standards, vendors and products. A key to SOA is the notion of a software service, which is a set of business and technical capabilities packaged as a service and made available for use outside application, division, and even organizational boundaries. SOA is an important element of the overall architecture at Federal Student Aid. The aim of the planned architecture is to package business functionality currently embedded in and repeated across multiple applications into software services and to leverage these services in future development efforts. This will allow Federal Student Aid to build an enterprise that will be responsive to changing needs, facilitate systems maintenance and facilitate higher reuse of existing IT assets.

- **Enterprise Service Bus:** An ESB is an integration architecture implemented by technologies found in a category of middleware infrastructure products usually based on web-services standards. The ESB technology provides foundational services for a SOA via an event-driven and Extenisble Markup Language (XML)-based messaging engine ("the bus").

- **Distributed (Tiered) Architecture:** A distributed architecture divides a solution into logical parts (tiers) based on the separation of concerns principle. For example, user interface features are contained within one tier, business functionality and rule enforcement is contained in a separate tier and data access functionality is contained in yet another separate tier. This style of architecture has been proven time and again in solutions of varying size and scope to promote greater flexibility, simplified maintenance and increased reusability of components, which is consistent with Federal Student Aid's architectural principles mentioned in Section 2.2.

2.5 Architectural Model Overview

The architecture model is designed to reflect Federal Student Aid's concept of a distributed, multi-tiered approach to developing technology solutions. Each tier in the model provides a unique set of technical capabilities that when combined will meet Federal Student Aid's objectives. The model is depicted in Figure 2-2 below:

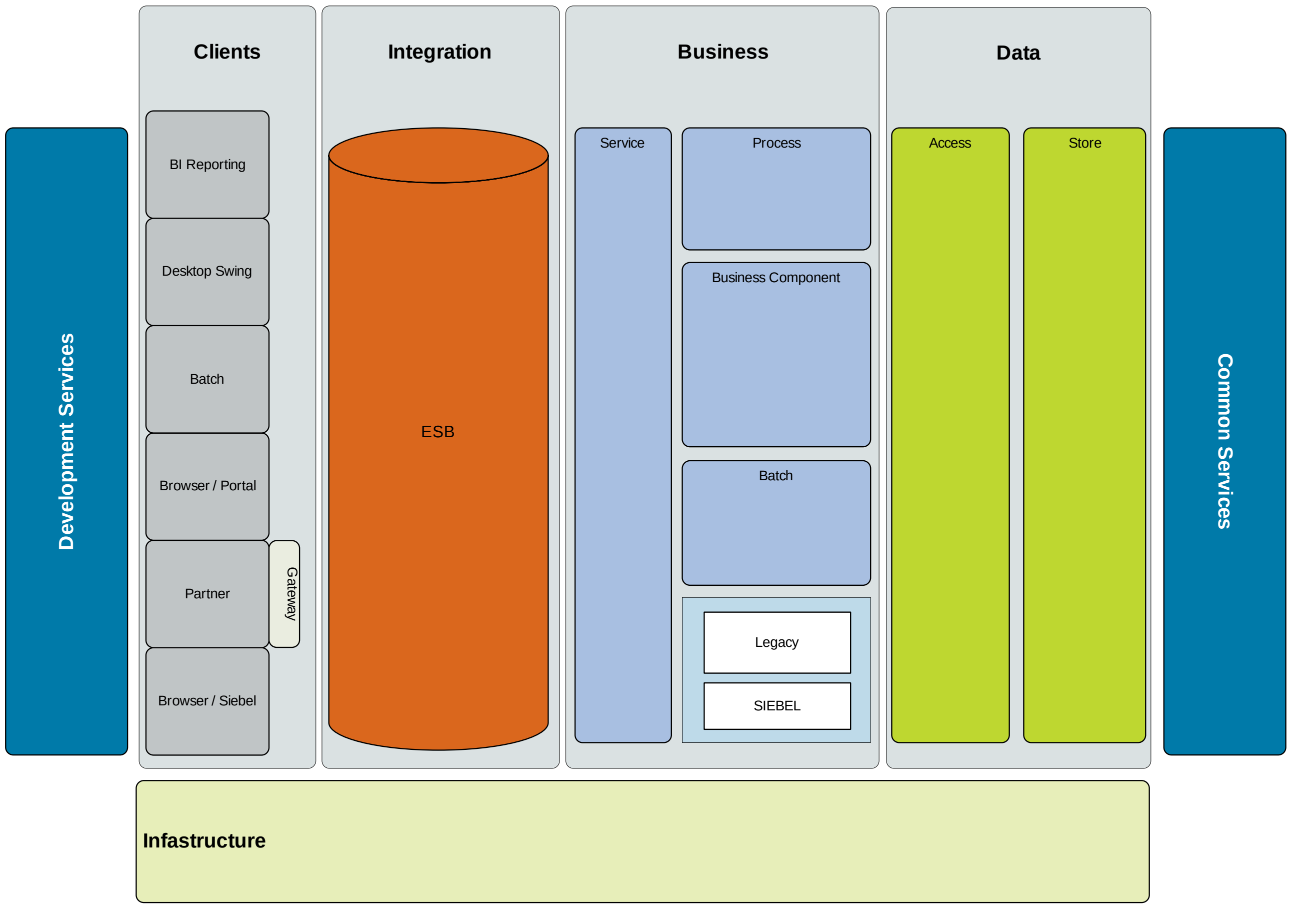

Figure 2-2: Federal Student Aid Architectural Model

2.6 Architectural Model Tiers

There are four major tiers in the architecture model. Each tier contains several technology elements that enable capabilities within that tier. In addition to the four tiers, there are three additional areas that do not play a direct role in delivery of business functionality but are necessary to implement each tier. These areas include development services, infrastructure, and common services. Their descriptions are outside the scope of this document. The relevant architectural tiers are presented below:

2.6.1 Client Tier

The Client tier of an application contains the types of users that will be interfacing with the applications. As identified in the Figure 2-2, the architectural model identifies six types of clients that can function as service consumers as appropriate:

- **Business Intelligence (BI) Reporting**: Enterprise reporting, querying and analytical solutions which can operate as standalone applications or be embedded within other applications;
- **Desktop Swing**: A Swing software application that is installed on an individual desktop computer;
- **Batch**: An application which initiates batch processing of business data;
- **Browser/Portal**: An application that is accessible from a web browser and that may or may not be deployed within an enterprise portal infrastructure;

- **Partner**: An application which interacts through a service, which may be the product of an external business partner or another government agency;

- **Browser/Siebel**: The user interface of a COTS software package, such as Siebel.

Detailed descriptions of these clients and their underlying technologies are outside the scope of this document because they are not considered part of the application architecture. These clients are listed to describe the types of clients that will exist in the Federal Student Aid technology environment. These clients can also be classified as service consumers, which will access Federal Student Aid services through the ESB.

2.6.2 Integration Tier

The Integration Tier, which consists of the ESB, will provide the integration services required to implement the SOA that will be used to integrate Federal Student Aid applications, services, and data. Accordingly, the ESB will provide the reliable and manageable integration services required to facilitate implementation and use of shared functions and to guarantee timely access to accurate shared data. Moreover, the ESB will bind islands of automation within the Federal Student Aid enterprise into an integrated target state business solution that will allow processes and data to be shared and coordinated with minimal restriction.

2.6.3 Business Tier

The Business tier implements the core business functionality and business logic of a solution and facilitates communication between the front end and back end tiers. At Federal Student Aid, the business tier will consist of transactional business components constructed to facilitate data processing. This tier also includes legacy and COTS products that are leveraged to provide application functionality, which may be exposed as services or called directly through external application programmatic interfaces (API).

For the purposes of the application architecture, all outward facing components will be exposed as services via the ESB. Outward facing components are those components whose services will be leveraged by external applications. Business components that are not exposed as external services, i.e. those that are only used within the context of other solution components will be designed to adhere to the service oriented programming model enabling them to be exposed to the enterprise, through the ESB, at a later time if required.

2.6.4 Data Tier

The purpose of the Data tier is to facilitate communication between business components and back-end data stores in a controlled manner. The Data tier is subdivided into two levels as follows:

- Access: Consists of all application components that communicate with a database(s);

- Data Stores: Consists of the database schemas with which a given application must interact.

In most situations the business component performs the following three tasks:

- Creates an instance of a related data access component;

- Uses the data access component's interface methods in order to communicate with the database;

- Destroys the instance of the data access component once the necessary output has been received, the necessary operation(s) performed or a thrown exception has been caught.

Section 3: Application Architecture

The application architecture model provides the framework and guidelines for the development of custom-built enterprise solutions for Federal Student Aid. The key enabler of the application architecture is the business tier, which consists of services, processes, and business components. The Federal Student Aid Architectural Model is shown below in Figure 3-1.

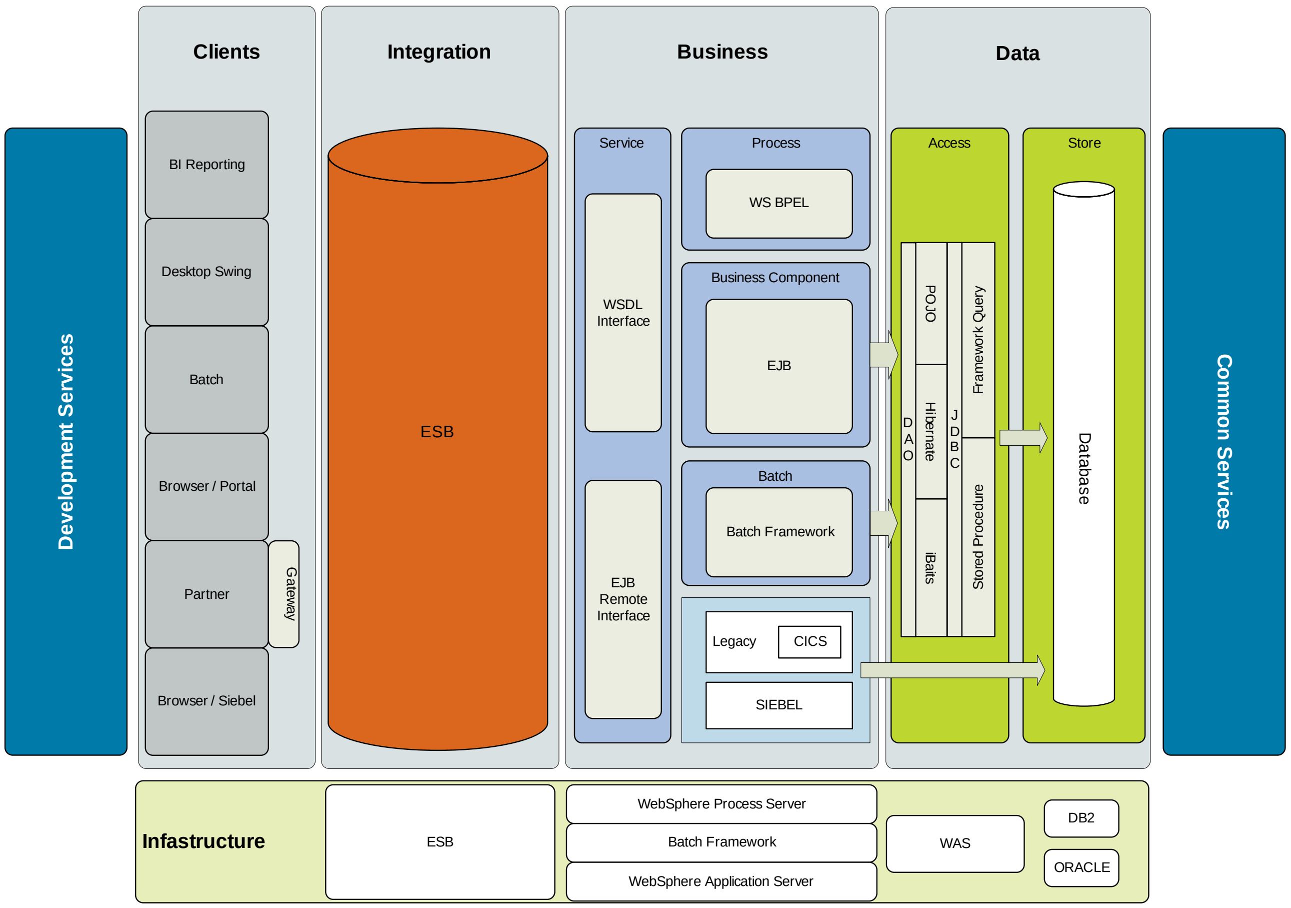

Figure 3-1: Federal Student Aid Architectural Model (static view)

The application architecture model has the following objectives:

- This application will support high volume transactional processing. Currently, the Central Processing System (CPS) processes hundreds of thousands of transactions on a peak day;
- This architecture will support batch processing. Currently, the Common Originations and Disbursements system (COD) manages thousands of batch files from schools participating in Federal Student Aid programs;
- This architecture will support distributed transactions. A distributed transaction is a transaction that is divided into threads of execution that spans multiple, independent, cooperating transactional systems or components. An example of a distributed transaction might involve a web application that must execute a logical business transaction by invoking a number of back-end business services that are deployed within independent application servers that each commit data to a persistent store. The single logical business transaction spans the multiple, independent application servers whose services are invoked;
- This architecture will support container managed transactions. A container managed transaction is implemented by an application server or other container or operating environment that hosts the execution of business logic. Typically, the implementation of the business logic, in Java or other programming languages, does not need to be concerned with explicit management of database transaction management;
- This architecture will support horizontal and vertical scaling, also known as "scaling up" and "scaling out." Scaling up refers to increasing available processing resources by adding Central Processing Units (CPU) (or cores) to a single server that supports a multi-CPU configuration. Scaling out refers to increasing processing power by adding additional servers to a cluster;
- This architecture will help realize the advantages of component and reusable service-based software by supporting service orientation.

In addition to the objectives outlined above, Federal Student Aid is interested in establishing a common Service Oriented programming model for all Java applications. Solution providers will be expected to design their solutions using Service Oriented Architecture design principles. It is expected that every solution will have a well defined services layer, component layer and Plain Old Java Object (POJO) layer regardless of its requirements to provide or consume services with the enterprise. Federal Student Aid is considering the adoption of the Service Component Architecture (SCA) as this Service Oriented Architecture programming model. SCA brings many benefits to the project team. These benefits include:

- Simplify the design and deployment of services;

- Establish a "transportable" set of engineering skills for service-oriented architecture (SOA) design;

- Enable many of the static analysis features that developers have come to expect in programming environments, but that have been absent in services (such as dependency analyses and type checking).

The remaining subsections present a static view of the model that identifies technologies and recommendations for building an application. Section 6 presents a dynamic view of the application model through the illustration of various use cases that would apply to the application model.

3.1 Business Tier

The business tier contains an application's services, business processes and business components that may consist of POJOs, Enterprise Java Beans (EJB) and BPEL flows. The business tier can be broken down into four major components:

- **Service**: Provides the interface and bindings to expose business functionality to service consumers;
- **Process**: Provides the high-level implementation of business processes and workflows. This is achieved by choreographing the performance of operations within related business services to achieve a business result.
- **Business Component**: Executes business logic and persists transactional data to the database.
- **Batch**: Facilitates efficient processing of large volumes of data. The batch component will be coupled with the business component to implement business logic for both transactional (single-record) and batch operations.

3.1.1 Service

The service component consists of the technologies that will be used to publish services to the ESB and the outside world. All components in the business tier should publish Web Service and Remote EJB interfaces.

Web Services Interface

- **Best Practice:** Web Services Definition Language (WSDL) standards are used to define service interfaces exposed to external applications.
- **Description:** A Web service is a software component that is designed to support machine to machine (consumer to provider) interaction over a network through standard web protocols, regardless of the implementation platforms of the consuming application and the service itself. The communication mechanism within web services is XML-based messaging that follows the SOAP-standard. Common in both the field and the terminology is the assumption that there is also a machine readable description of the methods (operations) and signatures (required inputs and outputs) provided by the service, in the WSDL file. A web service will not support the flow of a transaction context between the service consumer and provider (no two-phase commit support). The service itself can provide transactional support and optionally should support compensation in the event that rollback becomes necessary, most often as the result of an exception occurring during processing.
- **Rationale:** Services exposed with Web Service bindings help the Federal Student Aid architecture adhere to an SOA standard where the basic unit of communication is a message, rather than an operation. This is often referred to as "message-oriented" services. Additionally, designing services today with a standard WSDL interface will allow for easy deployment of services in an SCA environment that Federal Student Aid will most likely move to in the future. The key benefits are outline in Table 3-1 below:

Key Benefit	Architectural Fit
SCA Support	SCA facades can easily be programmed to leverage existing Web Service Interfaces. Additionally, SCA bindings can be used to synthesize web service bindings to provide

Key Benefit	Architectural Fit
	backwards compatibility with existing clients.
Industry Standard	SOA Web services are supported by most major software vendors and industry analysts. Web Services allow for communications between different implementation platforms (e.g. .NET and Java 2 Enterprise Edition (J2EE)). Web Services also promote loose coupling because the focus becomes the "contract" that the WSDL provides, rather than the underlying implementation details.

Table 3-1: Web Services Interface Key Benefits

- **Adoption Status:** Growing - Limited institutional experience at Federal Student Aid. This is a mature industry standard. Federal Student Aid is currently infesting in a robust ESB infrastructure to support SOA based solutions.

- **Related Standards and Technologies**: World Wide Web Consortium (W3C) Specification; SOAP; WSDL; Universal Description, Discovery and Integration (UDDI).

EJB Remote Interface

- **Best Practice:** EJB Remote Interfaces are the primary implementation technology for concert service interface implementations.

- **Description:** The EJB remote interface provides the actual access to the business-specific functionality of an EJB.

- **Rationale:** Designing services today with a standard EJB remote interface will allow for easy deployment of the EJB component in an SCA environment that Federal Student Aid will most likely move to in the future. EJB bindings should be used where transaction context is required to flow between the service consumer and provider (support of two-phase commit is required). EJB bindings should be used when the services needs to start or join a transaction where all activities within the transaction must commit or fail as a logical unit. SOAP based bindings should be considered first. If it is determined that compensation-based rollback is too complicated or impossible, EJB bindings should be considered. The key benefits are outline in Table 3-2 below:

Key Benefit	Architectural Fit
SCA Support	SCA facades can be built to leverage EJB remote interfaces.
Industry Standard	The EJB remote interface is the industry standard for remotely accessing an EJB in a pure Java environment.

Table 3-2: EJB Remote Interface Key Benefits

- **Adoption Status:** Emerging - No institutional experience at Federal Student Aid. This is a mature industry standard.

- **Related Standards and Technologies:** EJB Specification

3.1.2 Process

The process component consists of the technologies that will be used to orchestrate simple business components into more complex process flows.

Web Services Business Processing Execution Language (WS-BPEL)

- **Best Practice:** WS-BPEL supports the orchestration of business services to realize more complex processes and flows. Service providers will have WS-BPEL available for service development via WebSphere Process Server (WPS). WebSphere BPEL extensions may be used in Federal Student Aid solutions.

- **Description:** Process flows are implementations of workflows that choreograph several less complex business services to construct a complete business process. If one or more services fail to execute correctly, the business process flow understands the steps or actions necessary to handle exception conditions. It is important to note that many of the Federal Student Aid approved technologies include an orchestration tool e.g. FileNet, Siebel and Tivoli. When selecting the appropriate orchestration tool, vendors need to consider the business requirements. Federal Student Aid would like to standardize on WS-BPEL using the WebSphere Process server when there is no compelling reason to use the other workflow engines available in the infrastructure. FileNet provides unique capabilities to manage the workflow of documents, if these unique abilities meet the business needs better than process server, the alternate work flow engine should be used. Federal Student Aid recommends that the Process Server workflow engine be considered first and alternate workflow engines be justified against the inability of process server to meet the needs of the business requirements.

- **Rationale:** Federal Student Aid recommends WS-BPEL for implementing process services. WS-BPEL provides the capability to direct flow of information to various systems in the context of a business process. This will enable Federal Student Aid to better coordinate the activities of various enterprise applications that play a key role in enabling business processes. The key benefits are outline in Table 3-3 below:

Key Benefit	Architectural Fit
Scalability	WS-BPEL processes deployed in WebSphere Process Server scale with Process Server.
SCA Support	Full specifications exist for wrapping WS-BPEL processes in SCA facades both in WebSphere Process Server and via using Web Service Bindings.

Key Benefit	Architectural Fit
Industry Standard	WS-BPEL is an Organization for the Advancement of Structured Information Standards (OASIS) standard that is backed by Sun, IBM, HP, Adobe and many others.

Table 3-3: WS-BPEL Key Benefits

- **Adoption Status:** Emerging - No institutional experience at Federal Student Aid. This is an emerging industry standard.

- **Related Standards and Technologies:** WS-BPEL Specification, IBM Websphere Process Server

3.1.3 Business Component

The business component will executes business logic and persist transactional data to the database.

Stateless Session Beans

- **Best Practice:** Stateless session beans implement business components that need to manage transactions.

- **Description:** A session bean encapsulates business rules and logic pertaining to a defined operation or set of operations and is one of three types of components within the EJB specification (the other two types of beans being entity beans and message beans or message-driven beans). It is instantiated by a client, is interacted with through it's remote interface, and is reused by other clients when the processing is complete.

  A stateless session bean does not maintain a conversational state for a particular client. When a client invokes the method of a stateless bean, the bean's instance variables may contain a state, but only for the duration of the method invocation. In contrast, a stateful session bean maintains conversational state across methods and transactions and each instance is attributed to a specific client.

  Since stateless session beans can easily support multiple clients, they offer better scalability for high-traffic applications. However, as a trade-off, session state will need to be maintained in the Client tier using a state management mechanism that is applicable to type of client being used.

- **Rationale:** Stateless session beans have been selected as the key enabling technology for implementing business component functionality because they provide transaction management, SCA support, are highly scalable and are well accepted in industry. The key benefits are outline in Table 3-4 below:

Key Benefit	Architectural Fit
Transaction Management	Stateless session beans can manage complex transactions involving multiple resources, allowing transactional context to be transferred between components and processing layers of an application in order to maintain transactional integrity. Transaction management can be delegated to the EJB container or managed within the bean code.
Scalability	Stateless session beans are well-suited to high volume applications and are an industry-accepted means for implementing business functionality in enterprise applications. Existing technologies can easily support scaling through threading and clustering technologies.
SCA Support	Full specifications exist for wrapping EJB containers and stateless session beans in SCA facades. SCA will allow Federal Student Aid to evolve its solutions over time. There are a number of competing technologies to Stateless Session beans. Using the SCA programming model, solutions will be able to leverage these technologies over time without requiring massive redesign of existing solutions.
Industry Standard	Stateless session beans are currently a widely adopted industry standard used within n-tiered distributed applications. There is considerable debate in the industry concerning evolving component technologies and consequently we anticipate that EJB technologies may be replaced by other immerging technologies in time. This is why SCA is critical to this architecture. The application architecture used for Federal Student Aid solutions must be able to accommodate technological shifts over time without requiring massive redesign or restructuring of an application.

Table 3-4: Stateless Session Bean Key Benefits

- **Adoption Status:** Emerging - No institutional experience at Federal Student Aid. This is a mature industry technology.

- **Related Standards and Technologies:** EJB Specification, IBM Websphere Application Server

3.1.4 Batch Processing

A batch processing framework has not yet been selected at this time; however, the only constraint on a batch framework will be that the framework must implement business logic via remote instantiation of the stateless session bean business component. Key functional considerations for a batch framework include:

- Partitioning: Partitioning is a process of packaging and managing batch processing streams so as to maximize throughput and maximize utilization of available hardware resources while preserving the integrity of the underlying databases(s) and processes. Partitioning typically involves both division of a large batch stream into smaller units of work and combining small batch streams into larger units of work. The objective is that units of work have minimum dependencies upon each other so that they can be processed concurrently. Additionally, if batch transactions have any dependencies upon each other (e.g. one must be executed first before another one to be processed correctly or related transactions must produce some sort of "group" total or aggregate), those transactions would be placed into the same partition so that they are executed in the proper order and will properly compute any required aggregates.

- Scalability: The framework should be able to scale vertically and horizontally.

- Ability to manage checkpoints and restarts: Each component that processes a batch of data should save its state periodically so that in the event of a failure, it can be restarted and will be capable of carrying on where it left off. The checkpoint / restart must preserve the integrity of its results. In particular, it must not process any transaction more than once and must not skip any transactions. When concurrency and parallelism are added into the mix, the management becomes more complicated in that each process that is executing in parallel must be capable of performing a checkpoint / restart across all dependent processes.

Federal Student Aid has examined WebSphere Extended Deployment (XD) and recommends that solution providers investigate this tool when considering the architecture of solutions that require a batch-processing framework. Federal Student Aid has also discussed WebSphere Information Server as a base for a batch-processing framework. Federal Student Aid requires that solution provider's work closely with CIO personnel to collaboratively select the appropriate batch processing framework. Federal Student Aid does not recommend implementing transactional business logic in Information Server. All transactional business logic should reside in the EJB implemented in the business component layer.

- **Adoption Status:** Mature – Proven experience at Federal Student Aid

3.2 Data Tier

The sole purpose of the Data Tier of an application is to facilitate two-way communication between business components and back-end data stores in a controlled manner.

3.2.1 Data Access

Design Patterns

- **Best Practice:** The Data Access Object (DAO) design pattern will be the base design pattern used to develop data access components within applications. It may be used in conjunction with other design patterns as circumstances warrant.

  (See http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html for more information about the DAO pattern.)

- **Description:** The DAO-based data access component encapsulates all data operations and abstracts the underlying implementation specifics from the business component (EJB) that instantiates it. It is responsible for managing transactional state with the database, executing database calls, and preparing inputs and outputs – Value Objects -- into formats that business components can use and process.

- **Rationale:** The DAO design pattern is consistent with the architectural principles provided in Section 2.2 of this document. Specifically:

- This design approach has been proven effective in a wide variety of applications, from smaller applications with limited processing demands to high-demand enterprise applications;
- The encapsulation of data and abstraction of processing specifics promotes loose coupling of business components and data components, meaning that business components do not have to be developed based on the processing needs and requirements of the data access components;
- This approach is consistent with modular design and development practices, which facilitates the as-needed implementation of production-tested application components and centralization of maintenance;
- Value Objects can be returned to Business Components in multiple formats as necessary. Federal Student Aid has specified a desire to base all Value Objects on established XML schema standards for enterprise data where standards currently exist. In cases where enterprise-wide standards do not exist for a given business entity, the development team shall collaborate with Federal Student Aid's Data Management Group to define an appropriate standard;

The key benefits of Design Patterns are highlighted in Table 3-5:

Key Benefit	Architectural Fit
Scalability	These types of components have been deployed successfully in applications of all sizes.
SCA Support	Components can be utilized by business components wrapped in SCA. These components can also be wrapped in SCA and exposed as services themselves if necessary.
Industry Standard	The development of data access components built on DAO-based patterns and using Value Objects as transfer mechanisms is a proven practice that has wide acceptance within the industry. Furthermore, the abstraction of database communications in consistent with Federal Student Aid's desire to promote loose coupling of application components.

Table 3-5: Design Patterns Key Benefits

- **Adoption Status**: Emerging - No institutional experience at Federal Student Aid. This is a mature industry pattern.
- **Related Standards and Technologies**: No specific technologies. (See http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html for more information about the DAO pattern.) Readers are encouraged to search the Internet and review alternate DAO pattern descriptions.

Data Access Technologies

- **Best Practice**: Hibernate, iBATIS and Java Database Connectivity (JDBC) are recommended persistence frameworks to be used within data access components. The selection of the exact framework to be used is subject to development team's specific needs.

- **Description**: recommended applications of these technologies are described as follows:
  - Hibernate: Hibernate is an object-relational mapping (ORM) and query framework which translates data entities into concrete objects and provides complete abstraction of data access operations and value object generation, therefore requiring a relatively lower level of development skill when compared to iBATIS and JDBC. It utilizes a native query language and interface (Hibernate Query Language, or HQL for short) which is translated into Structured Query Language (SQL) to communicate with both relational and non-relational data sources;
  - iBATIS: iBATIS is a persistence framework that facilitates data access through a variety of means, including stored procedures and dynamic SQL and object-relational mapping through the use of SQL Maps, which are XML-based descriptors used to represent physical database objects or JDBC result sets. Similar to Hibernate, iBATIS provides an abstraction layer for database communications and can generate JDBC and SQL behind the scenes; however, iBATIS is more geared toward result set manipulation and its level of abstraction is somewhat lower, requiring comparatively more development skill than Hibernate but also allowing the developer to have slightly more control of communications and value object generation;
  - JDBC: JDBC is Java's standard mechanism to enable applications to communicate with both relational and non-relational data sources and is at the center of all data access frameworks. Direct coding of JDBC can be leveraged in environments where Hibernate and iBATIS are not used and where complete control of data access operations and value object generation is desired or required. The use of JDBC to communicate with the database will be limited to stored procedure calls (executed as java.sql.CallableStatement objects or equivalent based on the JDBC driver set used); at no time should embedded SQL statements be used. For more information on this decision, please refer to "Section 4 – Architectural Decisions."

- **Rationale**: All three frameworks can be effectively utilized within DAO-based data access component implementations, and it can be left up to the development team to select which framework(s) are to be used within a given project based on one or more of the following factors:
  - The type of data store(s) to be accessed;
  - The levels of control required in communicating with the database and generating value objects;
  - The skill mix and level of expertise of the development team.

The key benefits of Data Access Technologies are highlighted in Table 3-6:

Key Benefit	Architectural Fit
Transaction Management	All three frameworks have been proven to work in transactional environments.
Scalability	All of these frameworks have been leveraged in systems of

Key Benefit	Architectural Fit
	various sizes.
SCA Support	All three frameworks can be leveraged within an SCA environment, as they are utilized within the data access component itself.
Industry Standard	JDBC is Java's standard data access mechanism and is at the center of both Hibernate and iBATIS. The use of Hibernate and iBATIS is widely accepted within the industry. Furthermore, given the ability of Hibernate and iBATIS to abstract data access operations in varying degrees, they are consistent with Federal Student Aid's developer pool concept, whereby the reliance on specialized expertise for specific technologies can be reduced.

Table 3-6: Data Access Technologies Key Benefits

- **Adoption Status**: Emerging - No institutional experience at Federal Student Aid. These are mature industry technologies.
- **Related Standards and Technologies**: Hibernate; iBATIS; JDBC API

SQL Processing

- **Best Practice:** Database access will be performed through the use of database stored procedures or framework-generated SQL statements, such as those generated by iBATIS SQL Map descriptors and the Hibernate Query Language (HQL). As previously mentioned, no direct coding of SQL statements should be performed within data access components.

- **Description:** Stored procedures are developed within the database environment and provide a controlled interface through which data operations are conducted. Required inputs and outputs, along with their individual data types, are described in the call specification and data access components must be equipped to provide these items in order for calls to be successfully executed. Stored procedures are also useful for completing complex processing tasks, such as multi-table retrieve and update operations and data definition language (DDL) commands.

  Frameworks such as iBATIS and Hibernate provide abstraction layers which generate prepared SQL statements behind the scenes. This is particularly useful in situations where the development team lacks a strong database developer or administrator, or in situations where the target data store is not a relational database, such as XML documents, flat files and the like. In this instance, the application developer issues calls to exposed framework methods, which then use JDBC behind the scenes to prepare data store calls, receive and process outputs and return Value Objects, which are then utilized by business components. Development teams should plan to coordinate with the database administrator to optimize generated SQL statements and/or stored procedures to mitigate the risk of degraded system performance.

- **Rationale:** The use of stored procedures is ubiquitous in applications of every type and size, and predates the existence of the modern ORM frameworks. Most, if not all, application development teams will include database developers or administrators who develop and deploy stored procedure-based data access mechanisms as part of their core job functions. However, in situations where a development team does not possess strong database skills or significant productivity gains can be realized, the use of Hibernate and/or iBATIS can help to mitigate this skill gap or increase productivity.

The key benefits of SQL Processing are highlighted in Table 3-7:

Key Benefit	Architectural Fit
Transaction Management	These technologies can be utilized to perform both transactional and non-transactional data operations.
Scalability	Both stored procedures and queries generated by the Hibernate and iBATIS frameworks can be effectively used in applications of all sizes.
SCA Support	Not applicable, as these processes are embedded within the data access component and are abstracted from the business component layer.
Industry Standard	The use of stored procedures is a proven practice for developing database interfaces within applications and has wide industry acceptance. Dependent on the development team's skill set, the iBATIS and Hibernate frameworks can either leverage stored procedures or generate SQL and JDBC code automatically.

Table 3-7: SQL Processing Key Benefits

- **Adoption Status:** Emerging/Mature - No institutional experience in Hiberate and iBATIS at Federal Student Aid. Extensive institutional experience with JDBC at Federal Student Aid. These are mature industry technologies.

- **Related Standards and Technologies:** SQL (American National Standards Institute (ANSI))

3.3 Development Tools

Table 3-8 below identifies the tools that should be leveraged when developing and deploying the technology components that are part of the application architecture.

Key Development Tool	Technology	Notes
Application Server Toolkit V6.1	SCA on Websphere Application Server (WAS) 6.1 with SOA Feature Pack	There appears to be no automated generation of SCDL (Service Component Definition Language) files that contain the structured info used by the server to load and manage an SCA component. Text editors can be used to enter the SCDL XML (and possibly other artifacts) and manually place them in the project directory structure.
WebSphere Integration Developer	SCA on WebSphere Process Server	All SCA functionality is supported. A graphical component composition palette is provided. Automatic generation of required SCA SCDL and other required artifacts can be performed.
WebSphere Integration Developer	Business Process Execution Language Process Flows on WebSphere Process Server	WebSphere Integration Developer (WID) supports building BPEL flows between SCA components. The BPEL flows become new SCA components that can be used in new BPEL flows.
J2EE-Enabled Eclipse Integration Developer Environment (IDE) TOAD, SQL Navigator (or other database development client) ERWin, Rational Data Architect, Oracle Designer (or other data modeling tool)	Stateless Session Bean Database Development Database Design	Most open source tools can manage EJB development. Ensure that the selected tools can leverage WebSphere Application Server. These tools can be used to develop stored procedures, indexes and other database objects once a physical database model has been defined. Design tools can be leveraged to define key business entities, attributes and relationships (logical data models), which can then be used to generate physical database models (tables, fields, views, indexes, etc.)

Table 3-8: Key Development Tools

3.4 Adoption

Federal Student Aid has developed an adoption framework to help describe standard technologies as well as emerging ones that are under consideration for use. These technologies are listed in Table 3-6 below along with their associated adoption status:

- **Emerging**- No institutional experience at Federal Student Aid
- **Candidate**- No production experience but some experimental experience at Federal Student Aid
- **Growing**- Limited production experience at Federal Student Aid
- **Mature**- Proven production experience at Federal Student Aid

Technology	Adoption Status	Explanation
EJB	Emerging	No institutional experience at Federal Student Aid. This is a mature industry technology.

Technology	Adoption Status	Explanation
iBATIS	Emerging	No institutional experience at Federal Student Aid. This is a mature open source product.
Remote Method Invocation (RMI)	Emerging	No institutional experience at Federal Student Aid. This is a mature industry standard
SOAP	Growing	Some institutional experience at Federal Student Aid. This is a mature industry standard.
WS-BPEL	Emerging	No institutional experience at Federal Student Aid. This is a mature industry standard.
Java Messaging Service (JMS)	Candidate	Some production experience but some experimental experience at Federal Student Aid. This is a mature industry standard.
Stored Procedures	Mature	Several systems at Federal Student Aid use stored procedure extensively in both on-line and batch production

Table 3-9: Adoption Examples

3.5 Constraints

Federal Student Aid has standardized on several key technologies required to support Federal Student Aid's next generation application architecture. The following standards are core foundation technologies to this architecture:

- WebSphere Application Server (6.x +)
- WebSphere Process Server (6.x +)
- WebSphere MQ (6.x +)
- WebSphere Message Broker (6.x +)
- WebSphere Portal Server (6.x +)
- Oracle 10g Relational Database Management System
- Data Power XML Accelerator

The conceptual architecture assumes that these base technologies are available for use and focuses on implementation options and patterns.

Section 4: Architectural Decisions

This section presents key architectural decisions that were made as part of the architecture definition efforts. These decisions were derived from numerous discussions among key stakeholders over time. These decisions reflect current thinking of Federal Student Aid's for defining application architecture.

Architecture Decision			
1. **SQL-Java (SQLJ) is not a recommended technology**			
SQLJ, which stands for "SQL-Java[1]," is a multi-part specification for using SQL with Java: **Part 0**: Embedded SQL in Java. This provides a somewhat more object-oriented approach to the standard way of embedding SQL statements in programs. Part 0 supports static SQL statements in Java. It does not support dynamic SQL statements. Those are handled by JDBC. Part 0 does support mixing embedded static SQL statements with JDBC statements. Part 0 supports the same mapping between Java data types and SQL data types that is defined by JDBC. Also see the SQLJ execution environment (new window). **Part 1:** SQL routines using Java. This provides the ability to invoke methods written in Java from SQL code. The Java methods provide the implementation of SQL procedures. To support Part 1, a DBMS must have a Java Virtual Machine associated with it. Part 1 deals only with static methods. For an association between SQL functions and Java methods, each SQL parameter and its corresponding Java parameter must be able to be mapped and the two return types must be able to be mapped. Also see mapping SQL and Java data types (new window). **Part 2:** SQL types using Java. This defines SQL extensions for using Java classes as data types in SQL. Part 2 allows mapping of SQL:1999 User Defined Types (UDTs) to Java classes. It also allows importing a Java package into your SQL database by defining tables containing columns whose data type are specified to be a Java class. Structured types are associated with classes, attributes with fields, and initializers to constructors. All or part of an SQL type hierarchy can be represented in a Java class hierarchy. It is not necessary to associate the entire SQL type hierarchy to a Java hierarchy. Part 2 adds non-static methods to the static methods in Part1. Also see SQL:1999 (new window). (See http://www.service-architecture.com/database/articles/sqlj.html for full text)			
Architectural Area:	Data Tier	**Implications:**	JDBC and/or iBATIS and Hibernate should be used for data access instead of SQLJ.
Rationale:	SQLJ is considered somewhat of a niche technology and is not widely accepted in the industry.	**Decision Data:**	SQLJ is gradually being deprecated by many vendors, including Oracle, meaning that its

			long-term viability is questionable.

Architecture Decision			
2. Static SQL should not be coded inside of data access components			
Static SQL statements are hard coded in an application program when the source code is written. The source code is then processed using a SQL pre-compiler before it can be compiled and executed.			
Architectural Area:	Data Tier	**Implications:**	Stored procedures and framework generated queries are the only recommended mechanisms for data access.
Rationale:	Persistence frameworks can automatically generate SQL statements and execute stored procedures. As a result, developers do not need to know SQL in order to program calls to data stores. In the absence of a persistence framework, stored procedures can still be used to control access to business data the in underlying tables more effectively than can dynamic SQL queries (which in effect are not controlled).	**Decision Data:**	Given the ability of the persistence frameworks to generate SQL behind the scenes, regardless of whether or not stored procedures are used, there is no foreseeable need to hardcode SQL queries into data access code.

Section 5: Glossary and Standards

This section presents and defines key standards and technologies that are part of Federal Student Aid's application architecture model. The purpose of this section is to clearly reflect standards and technologies chosen for inclusion in the model. Readers should reference Appendix A for a complete list of acronyms used in this document.

The application architecture model is based on key concepts, standards and technologies that are part of the best practices for building modern business applications. However, some of these concepts, standards, and technologies are still evolving; therefore, they are presented here to avoid confusion and ambiguity on the interpretation of these concepts, standards, and technologies and their use.

Table 5-1 below presents a glossary of technologies and standards employed in this document.

Term	Definition
BI Tools	A suite of software tools used to provide standard and ad hoc reporting and analytical capabilities on an enterprise scale
CICS	Customer Information Control System (CICS) is a transaction processing system designed for both online and batch activity. CICS applications can be written in numerous programming languages, including Java, PL/I, C, C++, and IBM Basic Assembly Language.
COTS	Commercial off-the-shelf: A term for software or hardware products that are ready-made and available for sale, lease, or license to the general public. They are often used as alternatives to in-house developments or one-off government-funded developments.
CRM	Customer relationship management software
DAO	Data Access Object: A software component that provides a common interface between the application and one or more data storage devices, such as a database or file. Also, frequently used to refer to the Data Access Object design pattern.
DB2	DB2 is IBM's line of Relational Database Management System (RDBMS) (or, as IBM now calls it, data server) software products within IBM's broader Information Management software line. Most often DB2 refers to DB2 Enterprise Server Edition or the top of the line DB2 Data Warehouse Edition (DB2 DWE) which runs on Unix, Windows or Linux servers;
EAI	Enterprise Application Integration: Defined as the uses of software and computer systems architectural principles to integrate a set of enterprise computer applications.
EII	Enterprise Information Integration. It describes the process of using data abstraction to address the data access challenges associated with heterogeneous (federated) data
EJB	Enterprise Java Beans: A server-side component architecture used to develop transaction-based distributed systems on the J2EE platform
ESB	Enterprise Service Bus: An ESB is an integration architecture implemented by technologies found in a category of middleware infrastructure products usually based on

Term	Definition
	web-services standards. The ESB technology provides foundational services for a service-oriented architecture (SOAs) via an event-driven and XML-based messaging engine (“the bus”).
ETL	Extract, Transform, and Load: A process in data warehousing that involves * extracting data from outside sources, * transforming it to fit business needs, and ultimately * loading it into a data warehouse/data mart.
Hibernate	An ORM framework that can be used to transform data entities into concrete objects, and can be used to access both relational and non-relational database storage mechanisms. Hibernate is a public open – source framework.
HQL	The native query language used by Hibernate.
iBATIS	An ORM framework that can be used to leverage existing relational database investment and assets, including tables, views and stored procedures through the use of XML-based descriptors (SQL Maps)
JDBC	The standard data access mechanism provided with the Java platform, used to access a variety of data storage mechanisms such as relational databases, spreadsheets and flat files
JVM	Java Virtual Machine (JVM): A virtual machine that interprets and executes Java bytecode. This code is most often generated by Java language compilers, although the JVM can also be targeted by compilers of other languages.
LDAP	Lightweight Directory Access Protocol: An application protocol for querying and modifying directory services running over TCP/IP.
Legacy	Describes a system that is in current production use at the time a new development effort is undertaken
OLTP	Online Transaction Processing: A class of programs that facilitate and manage transaction-oriented applications, typically for data entry and retrieval transaction processing.
Oracle	An Oracle database consists of a collection of data managed by an Oracle database management system. Popular generic usage also uses the term to refer to the Oracle Database Management System (DBMS), but not necessarily to a specific database under its control.
ORM	Object-Relational Mapping: A programming technique for converting data between incompatible type systems in databases and Object-oriented programming languages.
POJO	Plain Old Java Object: A Java object which does not follow a specific object model, convention or framework and which, absent the use of any cross-platform integration technologies, can only be utilized by Java-based systems
Portal	A Web portal is a single point of access to information that is linked to various logically related internet-based applications and of interest to various types of users. Portals present information from diverse sources in a unified way. They provide an excellent way for enterprises to provide a consistent look and feel with access control and procedures for multiple applications, which otherwise would have been different entities altogether.
Query	A database query, the standard way information is extracted from databases.
RDBMS	Relational Database Management System: A database management system in which data is stored in tables and the relationship among the data is also stored in tables.
RMI	Java Remote Method Invocation is a Java application programming interface for performing the object equivalent of remote procedure calls. RMI allows Java objects running in different JVMs to communicate with each other.
SCA	Service Component Architecture: A set of specifications that describe a model for building applications and systems using a Service-Oriented Architecture. SCA extends

Term	Definition
	and complements prior approaches to implementing services, and SCA builds on open standards such as Web services.
Siebel	Siebel is principally engaged in the design, development, marketing and support of CRM applications. It is currently owned by Oracle
SOA	Service-Oriented Architecture: SOA is an architectural style. Applications built using an SOA style deliver coarse grained functionality as services that can be shared when building applications or when integrating within the enterprise or with trading partners.
SOAP	A technique for serializing object values to XML and reconstructing them as objects. SOAP handles the round trip between object and XML. May also be referred to as Service-Oriented Architecture Protocol
SQL	Structured Query Language: A fourth-generation (4GL) programming language used specifically to query a relational database and perform basic record operations such as reads, inserts, updates and deletes
Stored Procedure (Proc)	Stored Procedures: Subroutines and/or functions built within a relational database, which leverage SQL to perform database-level operations. Stored procedures are also used to provide applications with a well-defined interface and controlled access to underlying data
Swing	Swing is a graphical user interface (GUI) toolkit for Java. It is one part of the Java Foundation Classes (JFC). Swing includes GUI objects such as text boxes, buttons, split-panes, and tables.
Value Objects	Value Objects are a software design pattern used to transfer data between software application subsystems. Value Objects are often used in conjunction with Data Access Objects to represent data retrieved from a database.
WAS	IBM WebSphere Application Server: A software application server. WAS is built using open standards such as J2EE, XML, and Web Services.
Web Service	A software system designed to support interoperable Machine to Machine interaction over a network. Web services are frequently just Web APIs that can be accessed over a network, such as the Internet, and executed on a remote system hosting the requested services.
WebSphere Message Broker	IBM's information broker from the WebSphere product family that allows business data and information in the form of messages to flow between disparate applications across multiple hardware and software platforms. Business rules can be applied to the data flowing through the message broker to route, store, retrieve, and transform the information.
Websphere MQ	A network communication technology launched by IBM. It was previously known as MQSeries, which is a trademark that was rebranded by IBM in 2002 to join the suite of WebSphere products. WebSphere MQ is IBM's Message Oriented Middleware offering. It allows independent and potentially non-concurrent applications on a distributed system to communicate with each other.
WebSphere Portal Server	IBM's portal software which runs on top of WAS.
Workflow	A description of the movement of information through a work process
WS-BPEL	Web Services Business Process Execution Language: An XML-based programming language used to describe a business process
WSDL	Web Services Definition Language: An XML format used to define the external interface to a web service as a series of endpoints
XML	Extensible Markup Language: A platform-neutral markup language used to describe structured information in document form

Table 5-1: Glossary

Section 6: Use Cases

This section presents the architectural model in action. The dynamic representation of an architectural area will be accomplished by depicting various architectural components in use by a set of selected use cases. The actual representation of the dynamic model will be accomplished by developing sequence diagrams.

6.1 Overview

This subsection provides an overview of use cases applied to the application architecture model. The following use case will be used to describe this architecture and its components.

Loan Disbursement Component: The loan disbursement use case describes the steps involved to update the loan and disbursement databases when a new disbursement occurs. The loan disbursement is a key service that Federal Student Aid provides to its customers.

Note: This is a highly simplified example for illustrative purposes. It is not intended to represent the complexities of Federal Student Aid's production disbursement processing. It is primarily intended to illustrate how transactions can be managed and demarcated using the application architecture described in this model.

6.2 Loan Disbursement Component

When a loan disbursement occurs three key pieces of information are transmitted from the service consumer:

1. Borrower ID
2. Loan ID
3. Disbursement Transaction Information, in the form of a Value Object

The Borrower and Loan ID are queried against the person database to return the person and loan information that the disbursement applies to. This information is then combined with the Disbursement ID to create a disbursement record that is written to the database. As a final step the loan record is updated in the database with the new disbursement data. The sequence diagram for this use case is described in the following section.

6.2.1 Sequence Diagram

The Service Consumer initiates the process by sending the Borrower ID, Loan ID and Disbursement information as a Service Data Object (SDO) to the SCA runtime. The SCA runtime then transforms the SDO into a POJO that is sent to the Disbursement Component. The Disbursement Component is a stateless session bean residing in an EJB container, which encapsulates the logic necessary to make the disbursement updates within the context of a distributed transaction.

The following describes the interchange between the disbursement component designed to update disbursement information for a given person and loan. Process steps affecting each of the three concerned entities – Person, Loan and Disbursement – are provided:

<u>Step 1: Locate and lock the person record using the Borrower ID provided with the inbound SDO</u>

- The Disbursement Component EJB receives the inbound SDO and initiates a transaction within the called method;
- The Disbursement Component creates an instance of a Person DAO (PersonDAO) used to process operations on a Person record;
- PersonDAO opens a connection to the database, and notifies the instantiating business component if the connection attempt fails and the communication is terminated;
- Disbursement Component calls the requisite method of PersonDAO and provides the Borrower ID parameter to begin updating Person record information;
- PersonDAO accepts the inputs and calls the database to locate and lock the Person record for the duration of the transaction, in order to prevent potential data integrity issues resulting from multiple concurrent updates to the same record;
- If the lock is successful, PersonDAO receives notification from the database; if an exception occurs or no record is found the data access component throws an exception to the business component and the communication is terminated.

<u>Step 2: Locate and lock the loan record using the Person ID and Loan ID provided with the inbound SDO</u>

- The Disbursement Component creates an instance of a Loan data access object (LoanDAO) used to process operations on a Loan record;
- In order to connect to the database, LoanDAO either uses a reference to the existing connection within PersonDAO if connecting to the same database instance and schema, or creates its own database connection if connecting to a different one;
- Disbursement Component calls the requisite method of LoanDAO and provides the Person ID and Loan ID to begin updating the loan;
- LoanDAO accepts the inputs and calls the database to locate and lock the loan record;
- If the lock is successful, LoanDAO receives notification from the database; if an exception occurs or no record is found the data access component throws an exception to the business component and the communication is terminated. Any locks on Person and Loan are released and the transaction rollback logic within the Disbursement Component is invoked.

<u>Step 3: Insert the disbursement transaction to the database</u>

- The Disbursement Component creates an instance of a Disbursement data access object (DisbursementDAO) used to process disbursement operations;
- In order to connect to the database, DisbursementDAO either uses a reference to the existing connection within PersonDAO if connecting to the same database instance and schema, or creates its own database connection if connecting to a different one;

- Disbursement Component calls the requisite method of DisbursementDAO and provides the Person, Loan and Disbursement transaction data;
- DisbursementDAO accepts the inputs and calls the database to insert the disbursement;
- If the insert is successful, DisbursementDAO receives notification from the database; if an exception occurs, DisbursementDAO throws an exception to the business component and the communication is terminated.
- DAOs release any locks on Person and Loan records.

Step 4: EJB commits or rolls back the transaction and destroys DAO instances

- If the result from Step 3 was a successful processing of the operation, the EJB automatically commits the transaction and processing is returned to the calling object. If the EJB exits while throwing an error, the transaction is rolled back and processing is returned to the calling object.
- The "finally" block of the EJB method is processed regardless of transaction success or failure, where DAO instances are destroyed in the reverse of their order of creation -- DisbursementDAO is destroyed first, then LoanDAO and PersonDAO. Each DAO closes its database connection or returns it to the associated connection pool as it is destroyed.

Figures 6-1 and 6-2 illustrate the use case described above through the use of sequence diagrams. The first sequence diagram implements the data access objects using JDBC, the second uses Hibernate.

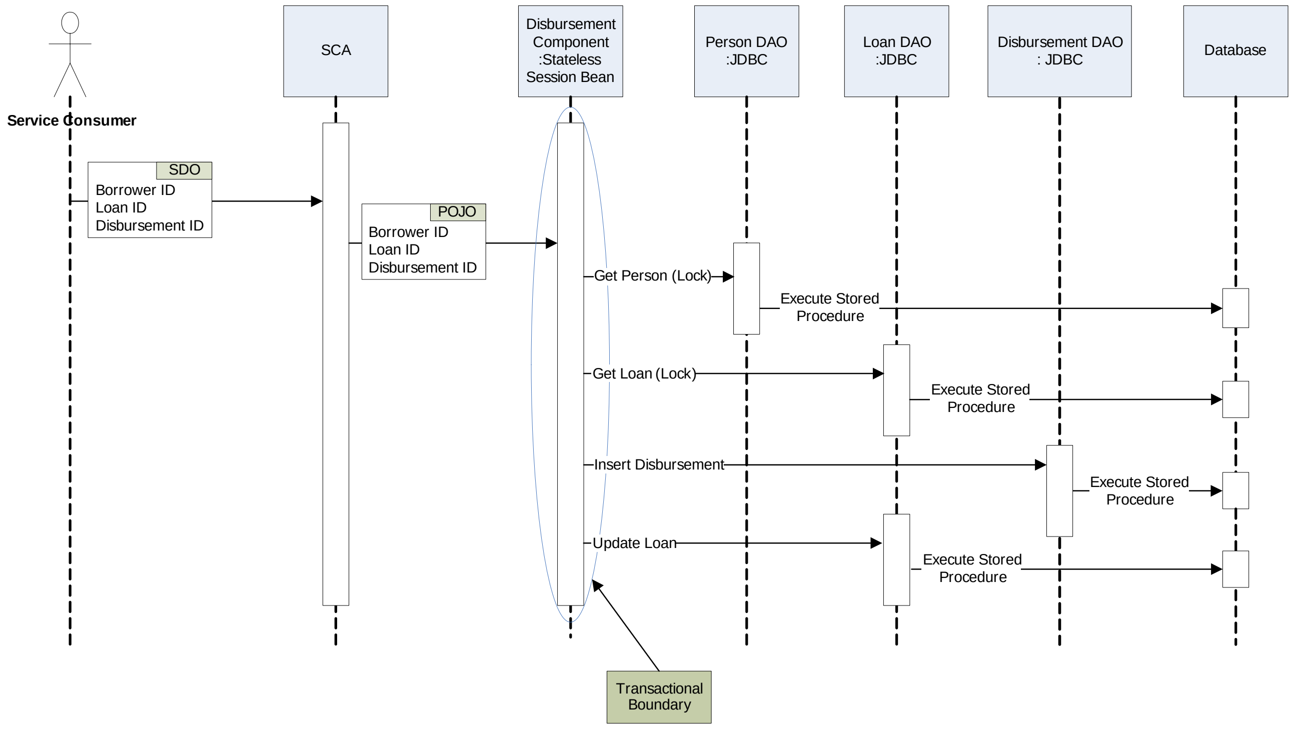

Figure 6-1: Loan Disbursement Component (JDBC) Sequence Diagram

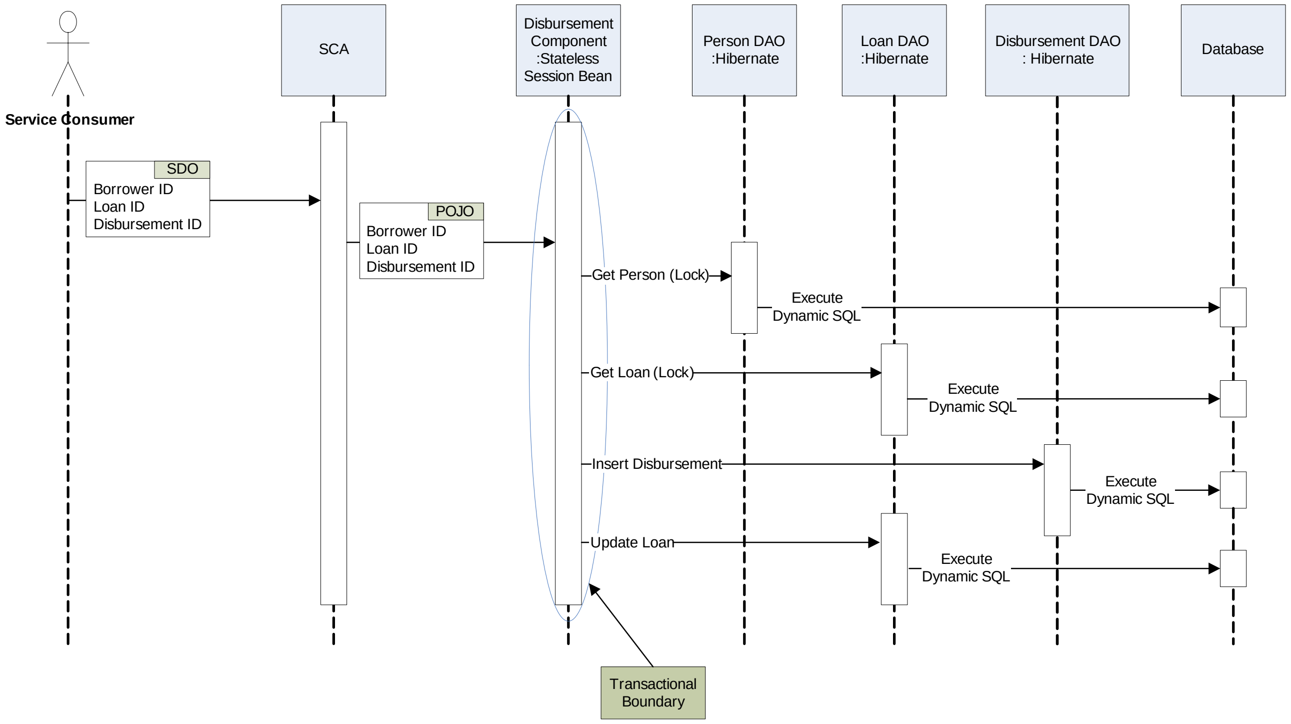

Figure 6-2: Loan Disbursement Component (Hibernate) Sequence Diagram

6.2.2 Transaction Boundaries and Locking

For the disbursement use case, pessimistic locking of the person and loan entity should occur. The transactional boundary will occur at the disbursement component. The application should leverage the intrinsic transaction boundary capabilities of the EJB and should be based on the needs of the individual application.

Appendix A: Acronyms

Acronym	Definition
ANSI	American National Standards Institute
API	Application Programmatic Interfaces
BI	Business Intelligence
BPEL	Business Processing Execution Language
CICS	Customer Information Control System
CIO	Chief Information Officer
COTS	Commercial Off the Shelf
CPS	Central Processing System
CPU	Central Processing Units
CRM	Customer Relationship Management
DAO	Data Access Object
DBMS	Database Management System
DDL	Data Definition Language
DWE	Data Warehouse Edition
EAI	Enterprise Application Integration
EII	Enterprise Information Integration
EJB	Enterprise Java Bean
EJBQL	Enterprise JavaBeans Query Language
ESB	Enterprise Service Bus
ETL	Extract Transform and Load
GUI	Graphical User Interface
HEA	Higher Education Act
HQL	Hibernate Query Language
HTTP	Hyper Text Transport Protocol
IDE	Integration Developer Environment
IF	Information Framework
IPM	Integrated Partner Management
J2EE	Java 2 Enterprise Edition
JDBC	Java Database Connectivity
JFC	Java Foundation Classes
JMS	Java Messaging Service
JVM	Java Virtual Machine
LDAP	Lightweight Directory Access Protocol

Acronym	Definition
NSLDS	National Student Loan Database System
OASIS	Organization for the Advancement of Structured Information Standards
OLTP	Online Transaction Processing
ORM	Object Relational Mapping
PBO	Performance Based Organization
POJO	Plain Old Java Object
RDBMS	Relational Database Management System
RMI	Remote Method Invocation
SCA	Service Component Architecture
SCDL	Service Component Definition Language
SDO	Service Data Object
SOA	Service Oriented Architecture
SQL	Structured Query Language
SQLJ	SQL-Java
TSV	Target State Vision
UDDI	Universal Description Discovery and Integration
W3C	World Wide Web Consortium
WAS	Websphere Application Server
WID	WebSphere Integration Developer
WPS	WebSphere Process Server

Table A-1: Acronym List